

Nom

Quizz #4: Rails

Q1 - How do you create a Rails app?

\$

Q2 - How do you start a Rails project?

1. Coding the views?
2. Coding the controllers?
3. Coding the models?

Q3 - How do you generate a `Song` model with a `title` and a `year`?

*Tell us the 2 files created and the **rake** command you should type then.*

\$

Q4 - How do you add a `category` (ex: `"rock"`, `"electro"`, etc..) to your model using the correct Rails generator?

*Again, what is the file created and the **rake** command you need to type again?*

\$

Q5 - Add a validation on presence of a song title + crash-test your model in the console

```
# models/song.rb
class Song < ActiveRecord::Base
  # Add the validation

end
```

Now crash-test your model:

\$ rails c
>

>
>
>
>
>
>
>

Q6 - What is the Rails flow you need to follow again and again? Give the correct order

1. The Router is routing the HTTP request to "controller#action"
2. The action is getting data from models
3. Everything starts with an HTTP Request
4. The action is rendering the view

Q7 - What are the **3 important parts** inside a HTTP request?

Q8 - Are these 2 routes the same? Why?

```
# config/routes.rb  
get "/songs" => "songs#show"  
post "/songs" => "songs#create"
```

Q9 - What's the difference between a **GET** and a **POST** request?

Q10 - What are the 7 CRUD routes generated by the `resources` method in Rails?

```
# config/routes.rb  
# TODO: Give us the details of the 7 routes generated  
resources :songs
```

Q11 - How do you print your routes and their URL prefix helpers?

\$

Q12 - How do you generate a controller for your songs?

\$

Q13 - Implement the **Read** actions in your songs controller?

```
class SongsController < ApplicationController
```

```
end
```

Q14 - What are the 2 requests needed to create a new song? Implement the `songs#new` and `songs#create` actions.

```
class SongsController < ApplicationController
```

```
  def new
```

```
  end
```

```
  def create
```

```
  end
```

```
  private
```

```
  def song_params
```

```
end  
end
```

Q15 - Why do we have to filter parameters using "strong params" in the controller?

Q16 - **Hard question:** What is the HTML generated?

Imagine that:

```
@restaurant = Song.new
```

Now what is the HTML code generated by:

```
<%= form_for @song do |f| %>  
  <%= f.text_field :title %>  
  <%= f.submit %>  
<% end %>
```

Fill the blanks:

```
<form action="          " method="post">  
  <input type="text" name="          " value="          ">  
  <input type="submit" value="Create song">  
</form>
```

Q17 - **Hard question:** What is the HTML generated?

Imagine that:

```
@restaurant # => <#Song: id: 18, title: "Hey jude", year: 1968, category: "rock">
```

Now what is the HTML code generated by:

```
<%= form_for @song do |f| %>  
  <%= f.text_field :title %>  
  <%= f.submit %>  
<% end %>
```

Fill the blanks:

```
<form action="          " method="patch">
```

```
<input type="text" name="          " value="          ">
<input type="submit" value="Create song">
</form>
```

Q18 - Now you want to add reviews to your app. Here are some constraints

- We don't want our visitors to destroy or update reviews, just to create ones.
- We don't want a separate index page to list all reviews or a show page to display each review. Instead, we want to display reviews on the show page of each song, this is a better UX.

Step #1: Model

Generate you `Review` model? It should have only a `content:string` and a `song:references` (= the foreign key).

```
$
```

Run the migration

```
$
```

Add validation/associations

- Add a validation for the presence of a review content
- Add associations between the two models

```
class Song < ActiveRecord::Base

end

class Review < ActiveRecord::Base

end
```

Step #2: Routing/Controller

Generate the reviews controller

```
$
```

An add the necessary routes (don't forget that we don't want the 7 CRUD actions for reviews)

```
# config/routes.rb
resources :songs do
  # TODO
end
```

Now code your controller

```
class ReviewsController < ApplicationController
  before_action :set_song

  def new

  end

  def create

  end

  end

  private
  def set_song
    @song = Song.find(params[:song_id])
  end
  def review_params
    params.require(:song).permit(:content)
  end
end
```

Step #3: Views

Add a song's reviews on its show page:

```
<h1><%= @song.title %></h1>
<p><%= @song.year %></p>
<p><%= @song.category %></p>

<h2>Here are the reviews for this song;</h2>
```

